Summarization Research Project

May 15, 2023

Mark Zhang

Part 1: Learning about Text Summarization

Current research in text summarization focuses on developing advanced deep learning models to generate concise and coherent summaries of text documents. Many researchers are exploring the use of transformer-based architectures, such as BERT and GPT, to improve the quality and fluency of generated summaries. Transfer learning and pre-training techniques are widely employed to leverage large-scale language models and enhance the performance of text summarization systems. Extractive summarization techniques, which select and merge key sentences from the source text, are being refined using various optimization strategies and sentense embeddings techniques. Abstractive summarization, where models generate new sentences that capture the essence of the original text, is an active area of research, aiming to produce more human-like summaries. Domain-specific summarization approaches are being developed to address the unique challenges of summarizing text from specific domains, such as scientific literature or legal documents. Evaluation of text summarization models is an important area of research, and various metrics have been proposed, such as ROUGE, BLEU, and BERTScore, to measure the quality of summaries. Nonetheless, evaluating the accuracy and the fluency of generated summaries remains a challenge, leading researchers to explore new evaluation metrics that are aligned with human preference.

Part 2: Implementing SOTA Methods

Introduction

In this section, I attempted to implement the current state-of-the-art abstractive summarization method based on the 2022 paper titled "Momentum Calibration for Text Generation." Although my implementation did not match the performance described in the paper, it did achieve significantly higher evaluation scores compared to the baseline.

Paper

The paper identifies two main issues with existing summarization techniques: the exposure bias problem and the loss-evaluation mismatch. The exposure bias refers to the model's tendency to predict the next token based on the previously predicted token, rather than the actual gold token during inference. The loss-evaluation mismatch occurs when there is a discrepancy between the token-level Maximum Likelihood Estimation (MLE) objective used during training and the sequence-level evaluation metrics used during inference. The authors propose a method that addresses these issues by introducing a momentum moving average generator model and an online model pair to align the model's loss with non-differentiable evaluation metrics.

Procedure

To implement the paper, I utilized Huggingface's Transformers and Datasets library, which provided high-level model loading and dataset manipulation functionalities. For the implementation of the custom loss function and training loop, I used the PyTorch library. The experiments were conducted on the NCSA hybrid cluster, utilizing the NVIDIA V100 32GB GPU. I applied the Momentum Calibration (MoCa) method to the finetuned BART model available on Huggingface (bart-large-cnn) and conducted a hyperparameter search, resulting in higher ROUGE scores on the CNN/DailyNews dataset.

Results

In comparison to the paper, my implementation yielded ROUGE scores that were 2% lower. This disparity may be attributed to the fact that the paper's author performed additional vanilla fine-tuning on the original BART model, which I did not have sufficient time to replicate. It is possible that with the same additional fine-tuning, I could achieve similar results.

Furthermore, the paper's author generated 16 candidate summaries per sample during training, whereas I generated only 8 to avoid exceeding GPU memory limitations. The larger set of candidate summaries may contribute to more stable convergence, as the variation in summarization could align the online model more accurately with the evaluation scores.

	Rouge 1	Rouge 2	Rouge L
Vanilla Finetuned Baseline in the MoCa Paper	44.22	21.22	41.01
MoCa Paper	48.88	24.94	45.76
Vanilla Finetuned Baseline in the BART paper	40.95	20.81	40.04
My Implementation	46.04	22.61	42.93

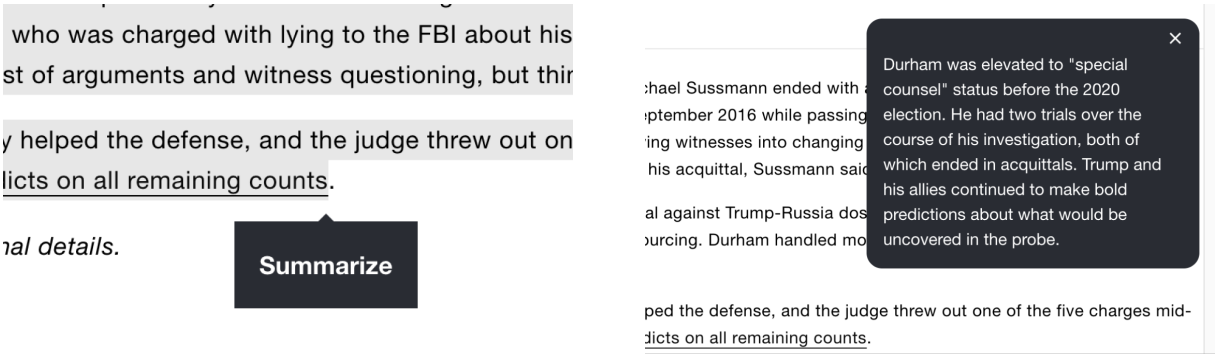
Observation

Although the paper outlines a momentum moving average technique for iteratively updating the generator model's weights, it does not specify the frequency of such updates. In my experiments, updating the model weights at each time step, as implied by the paper, did not yield satisfactory results. However, performing a single parameter update after approximately 16 steps with a batch size of 32 consistently produced the best outcome. Extended training duration and additional updates did not appear to enhance the evaluation score. Similarly, increasing the batch size and reducing the learning rate did not lead to further improvements in model performance.

Part 3: Application

Introduction

I have developed a Google Chrome extension that provides text summarization functionality for highlighted text. Users can select any text using their cursor, and a summarization button will appear. Upon clicking the button, a small floating window will display in the top right corner of the browser screen, showing a summary of the selected text.



Tech Stack

The Chrome extension is built using SolidJS, which enables easier control of state and reactive rendering. The backend utilizes Huggingface Transformers and Starlette. The summarization model employed is the BART model that I trained with MoCa on the CNN/DailyMail dataset. This design choice makes the Chrome extension particularly well-suited for summarizing online news articles.

Future Improvements

- Fine-tuning models on different datasets or having multiple models for each text type would enable the extension to accurately summarize various types of webpages.
- Implementing automatic text summarization on the page, without users having to manually select the text, would enhance convenience and user experience.
- Including citations for information extracted from the page and enabling users to access the source text by clicking on a sentence in the summary would promote transparency and facilitate further exploration.

Part 4: Reflection

Learnings

This exercise has consolidated my understanding of the mathematical concepts behind Language Models (LLMs) and their practical implementation in code. By delving into the source code of the Huggingface transformers library and developing a custom loss function using PyTorch, I was able to bridge gaps in my comprehension of LLMs. Additionally, the process of writing a custom training loop has equipped me with valuable skills in debugging CUDA errors and optimizing training efficiency. Implementing the specific techniques outlined in the MoCa paper has also deepened my knowledge of various text generation methods, such as diverse beam search and temperature sampling. Overall, this experience has proven immensely beneficial in preparing me for future endeavors in text summarization research.

Thoughts on Currrent Summarization Techniques

The current state-of-the-art models in text summarization have undoubtedly made significant progress and hold practical value. The model I trained in Part 2 demonstrates the ability to generate summaries of news articles that are generally satisfactory. However, it is important to acknowledge that these models are not flawless and there are areas where improvements can be made.

One notable limitation of current task-specific models is their relatively limited understanding of the underlying concepts conveyed in the text. While they can capture surface-level information and generate coherent summaries, they often lack the deeper comprehension required to accurately capture the nuances and intricate details of the original text. As a result, there are instances where the generated summaries may appear accurate at first glance but fall short in conveying the intended meaning or capturing the essence of the source material. This limitation becomes particularly evident when dealing with complex academic or technical texts, where precise language and domain-specific knowledge are crucial.

Thoughts on Application

- A platform that aggregates and summarizes user ratings and customer opinions from various websites and forums, providing a comprehensive overview of products sold online. This would enable users to quickly access key insights and make informed purchasing decisions without the need to extensively research multiple sources.
- An application that automatically generates summaries of podcasts and narrates them using the voice of the original speaker. This would allow users to quickly grasp the main points and highlights of podcast episodes, making it easier to decide which ones to listen to in full or to catch up on missed episodes.
- An application that automatically generates summaries for each episode of TV shows, enabling users to stay up to date with the plotlines and major events without having to watch every episode. This would facilitate engaging conversations with friends about the latest episodes, even if one doesn't have the time to watch all of them.